Operations Research is a multidisciplinary branch of mathematics involving mathematical modelling, mathematical optimization and statistical analysis in order to find "good" solutions for complex decision problems.
Typical objectives in OR are: maximize profit; minimize cost; minimize risk; minimize completion time; maximize efficiency, etc.

History of OR:
19th century—industrial revolution (efficiency of production processes)
World War II—birth of modern OR (Allocate scarce resources to various military operations in an effective manner)
1947 Simplex Algorithm 1953 Dynamic Programming
1970s Personal Computers —industry-size problems can be solved efficiently

Phases of an OR Study: (Course focuses on phases 2 and 3)
1.Define the problem of interest and gather relevant data.
2.Formulate a mathematical model to represent the problem.
3.Develop a computer-based procedure for deriving solutions to the problem from the model.
4.Test the model and refine as needed.
5.Prepare for the ongoing application of the model as prescribed by management.
6.Implement.

OR solves mathematical programming models

minimize $_x$ $z=f(x)$ subject to $x\in X$

$x\in R^n$ decision variables $f: R^n\to R$ objective function (e.g cost function)

$X\subseteq R^n$ feasible set (set of admissible decisions)

Any vector x that minimizes f is an optimal solution of the program and denoted by x^*. (might not unique or exist)
$z^*=f(x^*)$ optimal value achieved by x^*.

A **Linear Program** (LP) is a mathematical program that optimizes (maximizes/minimizes) a linear objective function over a feasible set described by linear equality and/or inequality constraints.
Note: LPs are much simpler to cope with than non-linear programs; CO477 - Computational Optimization deals with the theory of non-linear optimization.

Example: resource allocation problem,
Graphical Model—represent objective function as level lines.

Assume feasible set X bounded and nonempty, can prove that LPs have an optimal vertex solution.

Platonic Solid
a.Vertex optimal: single vertex optimal or both vertices optimal
b.LP infeasible (feasible set empty)
c.Remove constraints on availability of X and Y, and objective function can now grow to infinite on the feasible set. There is no maximum! The LP is unbounded.

LP as a tool for optimal decision making:
Maximize/minimize a linear objective function
Linear constraints (equalities and/or inequalities)
The feasible region is a convex polyhedron
The vertices of the feasible region contain a solution to the LP problem (if the LP is well-defined)
$\Rightarrow$ An LP can be solved by examining all vertices, but this approach is computationally prohibitive!

Standard Form

An LP is in **standard form** if:
—the aim is to minimize a linear objective function;
—all constraints are linear equality constraints;
—all constraints right hand sides are non-negative;
—all decision variables are non-negative. (continuous)

Minimize $z = c_1x_1 + c_2x_2 + ..+ c_nx_n$

Subject to
$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ..+ a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ..+ a_{2n}x_n &= b_2 \\ \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ..+ a_{mn}x_n &= b_m \end{aligned}$$
$x_1 \geq 0, x_2 \geq 0, .., x_n \geq 0$

The input parameters b_i, c_j and a_{ij} are fixed real constants that encode the LP problem.
Require $b_i \geq 0$, $\forall i=1..m$ (the decision variables x_i, i=1,..n are yet to be found.)

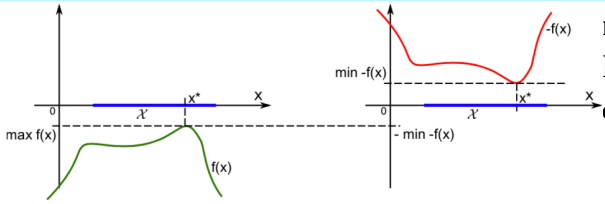
Compact Notation Ax=b where $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ and $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ and $x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ and $c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$

With matrix notation, the **LP in standard form** Minimize $z = c^T x$ Subject to $Ax = b$ $x \geq 0$, where b≥0.

Note: inequalities of the type x≥0 are understood to hold component-wise ($x_i \geq 0$, $\forall x_i \in x$)

General LP problems can
—be maximization (instead of minimization) problems;
—have inequality (instead of equality) constraints;
—have equality constraints with negative (instead of non-negative) right hand sides;
—have free (instead of non-negative) decision variables.
These general LPs can be transformed to standard LPs in a systematic way.

①Maximization→Minimization:



max y=f(x) s.t. x∈X → − min z=-f(x) s.t. x∈X
Inverting the objective preserves the optimal solution x*,
optimal value of the objective y*= − z*

②≤ Inequalities→Equalities: Slack Variables

Minimize $z = c_1x_1 + c_2x_2 + ..+ c_nx_n$

Subject to
$$\begin{aligned} a_{11}x_1 + ..+ a_{1n}x_n &\leq b_1 \\ \vdots & \vdots \\ a_{m1}x_1 + ..+ a_{mn}x_n &\leq b_m \\ \text{and } x_1 \geq 0, .., x_n \geq 0 & \end{aligned} \qquad \begin{aligned} a_{11}x_1 + ..+ a_{1n}x_n + s_1 &= b_1 \\ \vdots & \vdots \\ a_{m1}x_1 + ..+ a_{mn}x_n + s_m &= b_m \\ \text{and } x_1 \geq 0, .., x_n \geq 0 \text{ and } s_1 \geq 0, .., s_m \geq 0 & \end{aligned}$$

To reformulate ≤ inequalities as equalities, introduce m slack variables; after transformation, LP has n+m variables.

Matrix Notation— min $z=c^Tx$ | min $z=c^Tx$
s.t. $Ax \leq b$ | s.t. $Ax+s=b$ where $s=(s_1, .., s_m)^T$
$x \geq 0$ | $x \geq 0, s \geq 0$ (slack variables take the value of the difference b-Ax)

③≥ Inequalities→Equalities: Excess Variables

To reformulate ≥ inequalities as equalities, introduce m excess variables; after transformation, LP has n+m variables.

Matrix Notation min $z=c^Tx$ | min $z=c^Tx$
s.t. $Ax \geq b$ | s.t. $Ax-s=b$ where $e=(e_1, .., e_m)^T$
$x \geq 0$ | $x \geq 0, s \geq 0$ (excess variables take the value of the difference Ax-b)

Equivalence
—Assume initial problem not in standard form
—x: feasible solution to the initial problem
—(x, s): feasible solution to the standardized problem
—x can be associated with one and only one (x, s) using the reformulations we have defined.
—In particular, the optimal solutions will be x* and (x*, s*). x* will be the same in both formulations

④Negative Right Hand Sides:

If the right hand side of the i-th constraint is negative. If $b_i < 0$ in $a_{i1}x_1+a_{i2}x_2+...+a_{in}x_n=b_i$, then this constraint should be multiplied by -1. This yields $(-a_{i1})x_1 +(-a_{i2})x_2 +...+(-a_{in})x_n =-b_i$. The new constraint has a non-negative right hand side, we have $-b_i \geq 0$.

⑤Free Variables:

suppose there is no constraint $x_j \geq 0$, x_j can be positive or negative.

(Approach I) Substitute $x_j=x_j^+-x_j^-$ with $x_j^+, x_j^- \geq 0$. The LP has now (n + 1) variables: $x_1,.., x_{j-1}, x_j^+, x_j^-, x_{j+1},.., x_n$.

(Approach II) Any equality constraint involving x_j can be used to eliminate x_j.

Real-World LP Modelling

1.Resource Allocation: split a resource
- divide a valuable resource(capital, land, time..) among competing needs.
- variables: specify how much of the limited resource is allocated to each use
- constraints: express constraints on resource availability
- example: sharing CPU time (Find a share assignment that maximizes the total completion rate)

2.Blending Models: similar to resource models, but focused on combining resources
- decide what mix of ingredients best fulfills output requirements
- variables: specify how much of each ingredient to include in the mix
- constraints: express the composition of the output
- example: diet problem(Determine most economical diet, with basic nutritional requirements for good health)

3.Operations Planning Models:
- organizations must decide what to do, when and where (Manufacturing, distribution, government, ..)
- variables: multiple indexes identify products, types of activities, processing facilities, etc.
- constraints: balances between inputs and outputs of the activities
- example: transportation problem (satisfy shipping requirements and minimize total cost)

4.Shift Scheduling Models:
- focused on allocating workforce to tasks
- variables: typically, they indicate a number of employees
- constraints: allocate workers to cover activities

5.Time-Phased Models:
- used to address circumstances that vary over time.
- variables: express returns or state at given time
- constraints: time-phase balance constraints

6.others:

Scenario-based LPs: probabilities assigned to different possible situations

Feasibility problems: determine a subset of feasible constraints, for a given infeasible LP

Computational geometry problems: find the smallest convex polyhedron containing given points

Data envelopment analysis: measure the efficiency of decision making units

Solving LPs with GLPK:

GLPK is the official linear programming solver by GNU
- Free and open source (GPL license, written in C), Available on Linux, ported to Windows

GLPK implements several algorithms we see in class:

Linear Programming: Simplex algorithm Integer Linear Programming: Branch-and-Bound, Gomory cuts

GLPK is fairly similar to solvers used by OR professionals (Slower than commercial solvers. Language and features very similar to the AMPL+CPLEX commercial suite. AMPL+CPLEX is probably the most popular LP solver around)

Example: GMPL

1.Define a GMPL file specifying the linear program

—GMPL files are files in plain text with .mod extension

—Every mod file must be terminated by the keyword end;

—A mod file may include the keyword solve to explicitly request the solution of the optimization program.

—Comments are delimited either by /*...*/ or #
—GMPL syntax available in manuals

```
# example1.mod
var x{i in 1..2}, >=0; /* decision variables */
maximize y : x[1]+x [2];
s.t.
availX : 2*x[1]+x[2] <= 11;
availY : x[1]+3*x [2] <= 18;
demandA : x[1] <= 4;
solve ;
end ;
```

2.Solve the LP: glpsol -m example1.mod -o example1.out
Solution saved in example1.out, it is indeed vertex Q=(3,5)

Example: Shift Scheduling

A Police Department uses work shifts in which officers work 5 out of the 7 days of the week, with 2 successive days off. For example, a shift might work Sunday through Thursday and then have Friday and Saturday off. The following constraints are in place:
At least 6 officers must be on duty Monday, Tuesday, Wednesday, and Thursdays
At least 10 officers are required on Friday and Saturday
Exactly 8 officers are needed on Sunday
The Police Department wants to meet these staffing needs with the minimum total number of officers. For each day on duty an officer receives £100, except on Saturdays and Sundays where the pay is £80 in each day.
Assuming that the variables can be approximately treated as continuous quantities, formulate a shift scheduling program to minimize the cost for the Police Department.

The decision variables are
x_1 = number of officers working 5 days starting on Monday
x_2 = number of officers working 5 days starting on Tuesday
..
x_7 = number of officers working 5 days starting on Sunday
Linear programming formulation: min $z=500x_1+480(x_2+x_7)+460(x_3+x_4+x_5+x_6)$
subject to $x_1+x_4+x_5+x_6+x_7 \geq 6$ (Monday) $x_1+x_2+x_5+x_6+x_7 \geq 6$ (Tuesday) $x_1+x_2+x_3+x_6+x_7 \geq 6$ (Wednesday) $x_1+x_2+x_3+x_4+x_7 \geq 6$ (Thursday) $x_1+x_2+x_3+x_4+x_5 \geq 10$ (Friday) $x_2+x_3+x_4+x_5+x_6 \geq 10$ (Saturday) $x_3+x_4+x_5+x_6+x_7 = 8$ (Sunday)
$x_j \geq 0, j=1,...,7$

Standard Form

LPs in Standard Form:
$\min\ z = c^T x$ subject to $Ax = b \quad x \geq 0$ ($A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, where $b \geq 0$)
Assume, ①number of variables n ≥ number of equations m (otherwise, system Ax=b overdetermined)
②rows of A are linearly independent, rank(A)=m (otherwise, constraints are redundant or inconsistent)
Linear Dependence of rows in A implies either: ①contradictory constraints (no solution to Ax=b) x1+x2=1 x1+x2=2
②redundant constraints x1+x2=1 2x1+2x2=2

Index Sets

Consider only the system of linear equations in problem LP, Ax=b.
Let $A=[a_1, .., a_n]$, where $a_i \in \mathbb{R}^m$, is the i-th column vector of A.
Select a subset of m linearly independent columns a_i (always possible since m=rank(A) and n≥m).
Collect in the **Index Set I** the indexes for these m columns. I is therefore a subset of {1, . . . , n}.
The matrix $B=B(I) \in \mathbb{R}^{m \times m}$ consisting of the columns $\{a_i\}_{i \in I}$ is the **basis** corresponding to the index set I.

A solution x to Ax=b with $x_i=0$ for all $i \notin I$ is a **basic solution** (BS) to Ax = b with respect to the index set I .
A solution x satisfying both Ax=b and x≥0 is a **feasible solution** (FS).
A feasible solution which is also basic is a **basic feasible solution** (BFS).

The basic solution corresponding to I is unique.

As the vectors $\{a_i\}_{i\in I}$ are linearly independent, the basis B is invertible. Thus, the system $Bx_B=b$ has a unique solution $x_B=B^{-1}b \in \mathbb{R}^m$.
Define x = (x1,...,xn) through $(x_i)_{i\in I}=x_B$ and $(x_i)_{i\notin I}=0$. This x is the unique basic solution to Ax=b with respect to I.

Geometric intuition: LP solution at corner of feasible set
Algebra: Corners of feasible set correspond to basic feasible solutions

Example:

max y = x1 + x2	objective function
st. 2x1 + x2 ≤ 11	constraint on availability of X
x1 + 3x2 ≤ 18	constraint on availability of Y
x1 ≤ 4	constraint on demand of A
x1, x2 ≥ 0	non-negativity constraints

Vertices of the feasible set = basic feasible solutions!
Geometry: optimum always achieved at a vertex
Algebra: optimum always achieved at a BFS

Given an LP in standard form, a feasible solution to the constraints {Ax=b; x≥0} that achieves the optimal value of the objective function is called an optimal feasible solution. If the solution is basic then it is an optimal BFS.

Fundamental Theorem of LP: For an LP in standard form with rank(A)=m≤n:
∃ a feasible solution ⇒ ∃ a BFS ∃ an optimal solution ⇒ ∃ an optimal BFS.
The reverse is in general not true: there may be feasible solutions that are not BFS; there may be optimal solutions that are not BFS.

The naive statement "an LP has an optimal BFS" is also in general not true as the LP may be infeasible or unbounded.
Note: Theorem reduces solving an LP to searching over BFS's.

For an LP in standard form with n variables and m constraints, there are $\binom{n}{m}=\frac{n!}{m!(n-m)!}$ possibilities of selecting m columns in the A matrix. **There are at most (n m) basic solutions**: a finite number of possibilities!
Theorem 1 offers an obvious but terribly inefficient way of computing the optimum through a **finite search**.

There are (n m) index sets $I \subseteq \{1,...,n\}$ with |I| = m. But the number of distinct BFS is finite and usually < (n m), Because B(I) may be singular and the BS corresponding to I may not be feasible.

Fix an index set I with |I| = m and B(I) invertible.
The variables $\{xi\}_{i\in I}$ are basic variables(BV), while the variables $\{xi\}_{i\notin I}$ are nonbasic variables (NBV) corresponding to I.
Note: by construction, the nonbasic variables are always zero, but the basic variables can be zero or non-zero.

The basic representation corresponding to I is the (unique) reformulation of the system (z=cTx, Ax=b) which expresses the objective function value z and each BV as a linear function of the NBV's: $\begin{pmatrix} z \\ x_B \end{pmatrix} = f(x_N)$

where xB=[xi | i∈I] (BV's), xN=[xi | i∉I] (NBV's) and f: $\mathbb{R}n-m \to \mathbb{R}m+1$ is linear.

A linear program with multiple optimal basic feasible solutions:
max z = x + y s.t. x + y ≤ 2 x ≥ 0, y ≥ 0 (2 optimal BFS, multiple optimal non-BFS solutions)
All the points on the diagonal line (between the two optimal BFS) are optimal, but not basic. This can only happen when the gradient of the objective function is parallel to one of the constraint lines

Matrix Partition

Let A = [a1,...an], where $a_i \in \mathbb{R}^m$ is the i-th column of A. For any index set $I \subseteq \{1,...,n\}$ with |I| = m.

Define B=B(I)=[ai|i ∈ I], N=N(I)=[ai|i∉ I],
$c_B=c_B(I)=[c_i|\ i\in I]$, $c_N=c_N(I)=[c_i\ |i\notin I]$,
$x_B=x_B(I)=[x_i|\ i\in I]$, $x_N=x_N(I)=[x_i\ |i\notin I]$.

This implies $Ax = Bx_B + Nx_N$ $\quad c^T x = c_B^T x_B + c_N^T x_N$

Given this partition,

$\min\ z = c^T x$ $\quad$ $\min\ z = c_B^T x_B + c_N^T x_N$

subject to $Ax = b$ $\quad$ subject to $Bx_B = b - Nx_N$

$x \geq 0$ $(A\in \mathbb{R}^{m\times n}, b\in \mathbb{R}^m, c\in \mathbb{R}^n$, where b≥0)

Since B is invertible by construction, this implies $x_B = B^{-1}\left(b - Nx_N\right) = B^{-1}b - B^{-1}Nx_N$

Substitute into the expression for z $z = c_B^T x_B + c_N^T x_N = c_B^T B^{-1}b + \left(c_N^T - c_B^T B^{-1}N\right)x_N = c_B^T B^{-1}b + \left(c_N - N^T B^{-T} c_B\right)^T x_N$

The original system $z=c^Tx$, Ax=b is equivalent to the **Basic Representation**

$z = c_B^T B^{-1}b + \left(c_N - N^T B^{-T} c_B\right)^T x_N$

$x_B = B^{-1}\left(b - Nx_N\right) = B^{-1}b - B^{-1}Nx_N$ $\quad$ which expresses z and xB as linear functions of xN

Note: by setting x_N=0 we obtain the basic solution, $x = \left(x_B, x_N\right) = \left(B^{-1}b, 0\right)$ with **objective value** $z = c_B^T B^{-1}b$

Reduced Cost Vector: $r = c_N - N^T B^{-T} c_B$ (sensitivity of the objective function value z wrt the nonbasic variables x_N)

Importance of Basic Representations

Fix an index set I with |I| = m.

—Assume the basis B is invertible and the corresponding BS with $x_B = B^{-1}b$ and $x_N = 0$ is feasible (i.e. $B^{-1}b\geq 0$).

—The objective value of this BFS is $z = c_B^T B^{-1}b$.

—Any other feasible solution satisfies $x_N \geq 0$

—The basic representation $z = c_B^T B^{-1}b + r^T x_N$ and $x_B = B^{-1}b - B^{-1}Nx_N$ tells us how z and x_B change when the nonbasic variables increase.

In particular, the reduced cost vector r enables us to:

—recognize whether the current BFS is optimal (iff r ≥ 0, then no other feasible solution can have a lower objective value than the current BFS);

—find a new BFS with a lower objective value if the current BFS is not optimal (by increasing a nonbasic variable with a negative reduced cost).

Simplex Method

$\min\ z = c^T x$ $\quad$ subject to $Ax = b$ $\quad$ $(A\in \mathbb{R}^{m\times n}, b\in \mathbb{R}^m, c\in \mathbb{R}^n$, where b≥0)

Among the FS to LP, an important finite subset are the BFS.

A BFS is just the set of coordinates of a vertex. Thus at least one BFS will be optimal.

Each BFS is associated with a basic representation, i.e., a set of equations equivalent to z=cTx, Ax=b, that expresses the BV's in terms of the NBV's.

The basic representation also tells us the reduced costs, which indicate the best NBV to increase to improve the objective.

We can iteratively apply this idea until finding the optimal solution.

Basic Representation

$$\begin{aligned} z - \left(c_N - N^T B^{-T} c_B\right)^T x_N &= c_B^T B^{-1}b \\ x_B + B^{-1}Nx_N &= B^{-1}b \end{aligned}$$

Tableau

BV	z	$(x_B)^T$	$(x_N)^T$	RHS
z	1	0^T	$-r^T$	$(c_B)^T B^{-1}b$
x_B	0	I	$B^{-1}N$	$B^{-1}b$

where I is m×m identity matrix.

The tableau is a practical way to analyze the BS associated to the Basic Representation:

- RHS of the objective row is the objective value of the current BS
- RHS's of the other rows are the values of the basic variables at the current BS
- the coefficients of the nonbasic variables in the objective row are the negative reduced costs
- The current BS is feasible if and only if the RHS's are nonnegative in all rows (except objective row)

General Tableau Notation (for a feasible index set I with $p \in I$, $q \notin I$)

BV	z	x_1	..	x_p	..	x_q	..	x_n	RHS
z	1	β_1	..	β_p	..	β_q	..	β_n	β_0
:	:	:		:		:		:	:
x_p	0	y_{p1}	..	y_{pp}	..	y_{pq}	..	y_{pn}	y_{p0}
:	:	:		:		:		:	:

Note: $y_{ii}=1\ \forall i \in I$ and $y_{ji}=0\ \forall i \in I, j \in I\setminus\{i\}$ $\beta_i = -r_i\ \forall i \notin I, i \neq 0$ (negative reduced cost) $\beta_i = 0\ \forall i \in I$

Simplex Algorithm Idea:
if vertex x for index set I is not optimal, then in its neighbouring vertices there is one with a better objective value.
Neighbouring vertices are obtained by swapping a basic variable xp with a non-basic variable xq,
obtaining a new index set I'. xp leaves the basis, while xq enters the basis.
We use a technique called **pivoting** to efficiently compute the new basic representation for I' by updating I.
Similar to applying elementary row operations in Gaussian elimination. The pair (p, q) is the pivot
To swap xp and xq:

①divide row p by the pivot element y_{pq}, and relabel it as row q: $y'_{qj} = \frac{y_{pj}}{y_{pq}} \quad \forall j = 0,..,n$

②subtract row p multiplied by y_{iq}/y_{pq} from row $i \in I\setminus\{p\}$: $y'_{ij} = y_{ij} - \frac{y_{iq}}{y_{pq}} y_{pj} \quad \forall j = 0,..,n$

③subtract row p multiplied by β_q/y_{pq} from the objective row: $\beta'_j = \beta_j - \frac{\beta_q}{y_{pq}} y_{pj} \quad \forall j = 0,..,n$

Pivoting is possible if and only if $y_{pq} \neq 0$

Pivot Selection criterion
①**Feasibility**: new BS must be feasible; it must be a vertex. Thus we need to check in I′ that $y'_{i0} \geq 0, \forall i \in I'$.
(choose leaving variable xp).
②**Non-Inferiority**: new vertex must have a better objective value than the current vertex, $\beta 0' \leq \beta 0$.
(choose xq entering variable).

Choosing the variable xq which enters the basis
The objective row of the simplex tableau is equivalent to:
By definition, $\beta_i = 0$ for all $i \in I$ (basic variables), $\beta_i = -r_i$ for all $i \notin I$ (nonbasic variables).
In principle, any nonbasic x_i with $\beta_i > 0$ can enter the basis and be x_q, since each of these will decrease z.
—If there exist only a single x_i with $\beta_i > 0$, pick this as x_q.
—If several x_i have $\beta_i > 0$, pick the x_i with maximum β_i .
—If several x_i 's achieve the largest β_i, break the tie by picking the smallest index i.

Choosing the variable xp which leaves the basis
Once x_q enters the basis and it is set to a value $x_q > 0$, the constraints will force some basic variables to change value.

To ensure that all variables remain feasible, it must be $x_i = y_{i0} - y_{iq} x_q \geq 0 \quad \rightarrow \begin{cases} x_q \leq \overline{x}_{iq} \triangleq \frac{y_{i0}}{y_{iq}} & if\ y_{iq} \leq 0 \\ x_q \leq \overline{x}_{iq} \triangleq \infty & if\ y_{iq} \leq 0 \end{cases}$

The feasibility requirement is equivalent to asking that x_q is set so that these bounds are all simultaneously satisfied, $x_q \le \min_{i\in I} \bar{x}_{iq}$

①Suppose first that the bounds are trivial, $\min_{i\in I} \bar{x}_{iq} = \infty$, then

The entering variables x_q can grow indefinitely.
As $\beta_q>0$, the objective value $z=\beta_0-\beta_q x_q$ can drop indefinitely.
Then the LP is unbounded below and the optimization stops.
In this case, there's no need to choose the x_p variable.

②Suppose now that the bounds are non-trivial, $\min_{i\in I} \bar{x}_{iq} < \infty$,

The best value of the objective is obtained by making x_q as large as possible, we set $x_q = \min_{i\in I} \bar{x}_{iq}$,

Call p the row such that $x_{pq} = \min_{i\in I} x_{iq}$, i.e. the row that constraints the most the increase in value of x_q.
After p is chosen with this criterion, we are sure that feasibility is preserved if we set $x_q=\underline{x}_{pq} = y_{p0}/y_{pq}$
But then $x_p=y_{p0}-y_{pq} x_q=0$ becomes nonbasic. So this assignment defines a BFS, i.e., we moved to another vertex!
If there are several $p\in \arg\min_{i\in I} \underline{x}_{iq}$, we choose the one with the smallest index.

Simplex Algorithm (Minimisation)
Step 0: Find initial BFS and its basic representation.
Step 1: If $\beta_i\le 0$ for all $i\notin I$: STOP — the current BFS is optimal.
Step 2: If $\exists j\notin I$ with $\beta_j > 0$ and $y_{ij}\le 0$ for all $i\in I$: STOP — no finite minimum exists.
Step 3: Choose xq with largest $\beta_q>0$ Entry criterion — x_q enters the basis.
Step 4: Choose $p\in \arg\min_{i\in I} x_{iq}$ Exit criterion — x_p leaves the basis.
Step 5: Pivot on y_{pq} and go back to Step 1.

Maximum number of iterations for the standard simplex algorithm for a general problem with m rows and n columns: n variables in the index set so the max number of cycles would be checking every index set of size m until find the optimal one $C_m^n = \binom{n}{m} = \frac{n!}{m!(n-m)!}$

Significance of Klee-Minty Cube: Klee–Minty cube is an example that shows the worst-case computational complexity of many algorithms of linear optimization. It is a deformed cube with exactly 2^D corners in dimension D. Klee and Minty showed that Dantzig's simplex algorithm visits all corners of a (perturbed) cube in dimension D in the worst case.

Finite Termination and Degeneracy

A Basic Solution(BS) is **Degenerate** if one or more basic variables (BVs) are zero.
A degenerate BS has more than n−m zero-valued variables.
For tableau, there exists at least a BV such that $i\in I$ and $y_{i0}=0$.
A BS is called non-degenerate if all of its basic variables are different from zero.
Finite Termination: If all BFS's are non-degenerate, then the simplex algorithm must terminate after a finite number of steps with either an optimal solution OR a proof that the problem is unbounded.
Proof:
—At each step we have $y_{i0}>0$, $\forall i\in I$(non-degeneracy).
—Unless optimality (all $\beta_i\le 0$) or unboundedness is detected in STEP 1 or 2, we find $\beta'_0=\beta_0-(\beta_q/y_{pq})y_{p0}<\beta_0$
—Thus, the sequence of objective values obtained by the algorithm is strictly decreasing, $\beta_0>\beta'_0>\beta''_0>..$ No basic solution will ever be repeated.
—There are ≤(n m) basic solutions, since (n m) is the number of ways of picking m columns out of n to form an index set I.
—Thus, the process cannot continue indefinitely and must terminate at STEP 1 or 2 after a finite number of iterations (even though possibly a very large one!).

Assume that, $\forall i=1,..n, \exists$ BS $\underline{x}$ with $\underline{x}_i\neq 0$. Then, a BS x is degenerate if and only if it is associated with more than one index set.
Proof: BS x degenerate ⇐ BS x has more than one index set

—Suppose a BS x corresponds to index sets I_1 and I_2, $I_1 \neq I_2$.
—Then $x_i=0$ for all NBVs x_i with either $i \notin I_1$ or $i \notin I_2$ or both.
—In particular, since $I_1 \neq I_2$, there will be a NBV x_i in I_1, that is a BV in I_2. Since the two index sets describe the same BS x, x_i must be zero also in I_2 where it is basic.
—x is a degenerate BS. (The same holds for any x_i that is NBV in I_2 and BV in I_1)
Proof: BS x degenerate $\Rightarrow$ BS x has more than one index set
—Suppose x is a degenerate BS associated with some index set I; consider the corresponding simplex tableau.
—Due to degeneracy, $\exists p \in I$ with $y_{p0}=0$.
—$\exists q \notin I$ such that $y_{pq} \neq 0$. Otherwise, it would be always $x_p=0$ in all the feasible set which we assume impossible in the theorem statement.
—Pivoting on (p, q) gives a new basic solution which is identical to the current one since
—$y'_{q0}=y_{p0}/y_{pq}=0=y_{p0}$ and $y'_{i0}=y_{i0}-(y_{iq}/y_{pq})y_{p0}=y_{i0}$ $\forall i \in I \setminus \{p\}$.
—x corresponds to the index sets I and $(I \setminus \{p\}) \cup \{q\}$.

The index sets I and $(I \setminus \{p\}) \cup \{q\}$ produce the same BFS but different basic representations.
If we pivot on (p,q) when $y_{p0}=0$, then the new BFS is identical to the old one.
In particular, we find $\beta 0' = \beta 0 - (\beta q / yp0) ypq = \beta 0$, and the finite termination theorem breaks down (no strict improvement of objective value).
A pivot step (p, q) is called degenerate if $y_{p0}=0$ and non-degenerate otherwise.

The simplex algorithm can now be decomposed into:
[sequence of degenerate pivots] [non-degenerate pivot] [sequence of degenerate pivots] ..
Note: Some or all of these sequences of degenerate pivots may be empty.
Geometrically, the current BFS remains unchanged throughout a sequence of degenerate pivots, and a non-degenerate pivot moves it to a different BFS.
We know that the number of index sets is $\leq (n\ m)$.
$\Rightarrow$ Sequences of degenerate pivots are finite if no index set is repeated.
However, on same rare instances pivoting can result in a cycling behaviour.
In general, choosing degenerate pivots is a necessary condition, but not a sufficient one, for cycling.
After a sequence of pivots, we return to the same index set and so the algorithm will cycle and never terminate!

Bland's Rule (avoid cycling)
(i)Choose the lowest-numbered (leftmost) nonbasic column q with a positive cost. $q=\min\{j \neq 0 \mid \beta j > 0\}$
(ii)Denote as p the row with minimal $\underline{x}_{iq}$, in case of ties pick the row with the smallest index (same as standard conventions).
With Bland's rule the simplex algorithm cannot cycle and hence is finite.

Degeneracy in Practice:
Cycling was thought to occur in contrived examples. For a long time it has therefore been ignored in commercial solvers.
More recent experience with larger and larger problems indicates that cycling occurs, but it is still a rare event.
Rigorous remedies such as Bland's rule are not satisfactory as they increase the number of iterations and the work per iteration also in problems where cycling does not occur.
In practice it is acceptable to replace a $y_{i0}=0$ by $y_{i0}=\varepsilon > 0$ (e.g., $\varepsilon < 10^{-3}$) and then continue.

Two-Phase Simplex

Initial Basic Feasible Solution
— in Step 0, the simplex algorithm requires an initial BFS and the corresponding basic representation
— finding a feasible solution is in general as hard as finding an optimal solution!!!
— construct an initial BFS
in general, an initial BFS can be found using a variant of the simplex algorithm
in some special cases, an initial BFS can be constructed "manually"

All slack Basis

$$\min\ z = c_1x_1 + c_2x_2 + .. c_nx_n$$
$$\text{subject to}\quad \begin{matrix} a_{11}x_1 + a_{12}x_2 + .. a_{1n}x_n \le b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + .. a_{mn}x_n \le b_m \\ x_1 \ge 0, x_2 \ge 0 .., x_n \ge 0 \end{matrix}$$

$$\min\ z = c_1x_1 + c_2x_2 + .. c_nx_n$$
$$\text{subject to}\quad \begin{matrix} a_{11}x_1 + a_{12}x_2 + .. a_{1n}x_n + x_{n+1} \le b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + .. a_{mn}x_n + x_{n+m} \le b_m \\ x_1 \ge 0 .., x_n \ge 0, x_{n+1} \ge 0 .., x_{n+m} \ge 0 \end{matrix}$$

This is a basic representation for I={n+1, .. n+m}. The corresponding BS is feasible if $b_i \ge 0$ (i=1..m)

Example without an Obvious Initial BFS: ≥ inequalities & equalities
—only slack variables behave like basic variables!!
—add new artificial variables to those constraints that were originally equalities and ≥ inequalities!
—artificial variables behave like basic variables, now found a basic feasible representation
—But this system is not equivalent to the original one.

Phase 1:
Step 1: Modify the constraints so that all RHS's are nonnegative (constraints with negative RHS × – 1).
Step 2: Identify now all equality and ≥ constraints. In Step 4 we will add artificial variables to these constraints.
Step 3: Standardize inequalities: for ≤ constraints, add slacks; for ≥ constraints, subtract excesses.
Step 4: Add now artificial variables ξ_i to all ≥ or equality constraints identified in Step 2.
Step 5: Let ζ be the sum of all artificial variables and derive the basic representation for ζ.
Step 6: Find minimum value of ζ using the simplex algorithm.

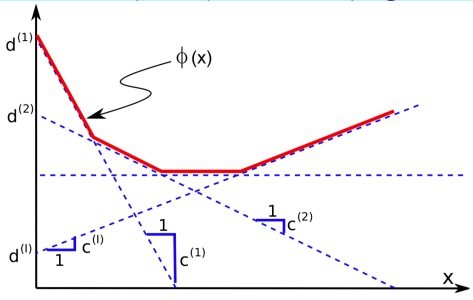
Phase 2:
Case ①: $\zeta^* > 0 \Rightarrow$ The original LP is infeasible.
Case ②: $\zeta^* = 0$ and all ξ_i are nonbasic at optimality
- Remove all artificial columns from the optimal Phase 1 tableau.
- Derive the basic representation for z (original objective) w.r.t. optimal index set of Phase 1.
- Solve the original LP with the simplex algorithm (Phase 2). The final basis of Phase 1 is the initial basis of Phase 2. The optimal solution to Phase 2 is the optimal solution to the original LP.

Case ③: $\zeta^* = 0$ and at least one ξ_i is basic at optimality
- As $\zeta^* = 0$ we conclude that all $\xi_i = 0$, thus some basic variables are zero.
- We have found a degenerate BFS for the original problem and a basic representation for the auxiliary problem.
- As the BFS is degenerate, we can pivot on a $y_{pq} \ne 0$ corresponding to an artificial ξ_q and an original variable x_q while keeping $\zeta^* = 0$!
- All ξ_i variables can thus be removed from the basis obtaining a feasible BFS for the original LP.

Linearization Methods

Linear programming is useful, but often fairly restrictive.
Nonlinear optimization problems arise very commonly in applications. (Euclidean distance, chemical reactions, queueing problems..) Nonlinearities can affect objective, constraints, or both.
Example: $z=(x_1)^2$, $z=x \bmod 2$,… $|x_1-x_2| \le 3$, $x_1x_2+x_3 \le 5$,...
Nonlinear programs can sometimes be **reformulated** as a single LP problem or as a sequence of LP problems.

Min-Max(MM) Models (important for game theory)



Consider a family of linear functions $y_i(x)=c_i^Tx+d_i$,
set $\phi(x)=\max_{i=1..I}\{c_i^Tx+d_i\}$ for $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$
Then, min $\phi(x)$
subject to $Ax=b$, $x \ge 0$ is **min-max problem**.

Theorem: Consider the following linear program: min z subject to $z \ge c_i^Tx+d_i$ ($\forall i=1,\ldots,I$) $Ax=b$ $x \ge 0$, z free

If (x^*_{LP}, z^*_{LP}) is an optimal solution of this LP, then x^*_{LP} is also an optimal solution of MM, and MM has optimal value $\phi(x^*_{LP}) = z^*_{LP}$. (Min-Max problems can be solved by LPs!!!)

Proof by contradiction:

—Let (x_{*LP}, z_{*LP}) be optimal in LP. x_{*LP} is feasible in MM since in LP it is $Ax_{*LP}=b$ and $x_{*LP}\geq 0$

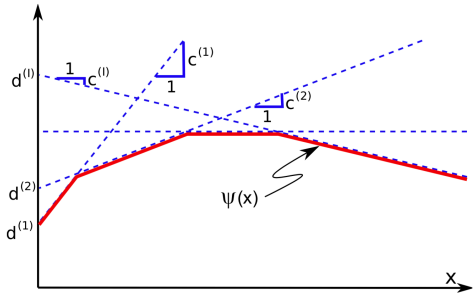
—Assume that x_{*LP}, is not optimal in MM, then there exist a x_{*MM} such that $\varphi(x_{*MM})<\varphi(x_{*LP})$

—Then $(x_{*MM},\varphi(x_{*MM}))$ must be feasible in LP, since in MM, we require $Ax_{*MM}=b$ and $x_{*MM}\geq 0$, and for all i=1..l

$$\phi\left(x^*_{MM}\right)=\max_{j=1,..l}\left\{c(j)^T x^*_{MM}+d(j)\right\}\geq c(i)^T x^*_{MM}+d(i)$$

—By the constraints in LP $z^*_{LP}\geq \max_{i=1,..l}\left\{c(i)^T x^*_{LP}+d(i)\right\}=\phi\left(x^*_{LP}\right)$ and since $z^*_{LP}\geq\phi\left(x^*_{LP}\right)>\phi\left(x^*_{MM}\right)$, x_{*MM} would achieve in LP a better objective than z_{*LP}. Thus x_{*LP} must be optimal also for MM.

Min-Min(MM') Models



Set $\psi(x)=\min_{i=1,..I}\{c_i^T x+d_i\}$ for $c_i\in\mathbb{R}^n$, $d_i\in\mathbb{R}$.

Then, min $\psi(x)$ subject to $Ax=b$, $x\geq 0$ is called a min-min problem.

Theorem:

Consider the following linear programs.

min $z_i=c_i^T x_i+d_i$

s.t. $Ax_i=b$ (LP(i))

$x_i\geq 0$

Let z_i^* be the optimal solution to LP(i). Let LP(j) be the LP that has the minimal objective, $z_j^*=\min_{i=1,...,I} z_i$ and let $x^*(j)$ be its optimal solution. Then $x^*(j)$ is optimal in MM' and $\psi^*=\psi(x^*(j))=z_j^*$.

Proof: The interchangeability of Min-Operators implied.

Interchangeability of Min-Operations: Let X andY be arbitrary sets, and let f: $X\times Y\to\mathbb{R}$ be an arbitrary function defined on $X\times Y$. Then $\min_{y\in Y}\min_{x\in X} f(x, y) = \min_{y\in Y}\min_{x\in X} f(x, y)$

Goal Programming

Minimize the sum of deviations of multiple linear objectives $c_i^T x$ from their respective targets d_i (i=1,..I)

$\min\sum_{i=1}^{I}\left|c_i^T x-d_i\right|$ The goal program can be solved as LP: $\min\sum_{i=1}^{I} z_i^+ + z_i^-$ s.t. $c_i^T x-d_i=z_i^+-z_i^-$ and $z_i^+, z_i^-\geq 0$

Rationale: Since $y_i=c_i^T x-d_i=z^+-z^-$ and $z^+\geq z^-$, y is positive; $z^+<z^-$, y is negative;
$|y|=|z^+-z^-|=z^-$ ($z^+=0$) or z^+ ($z^-=0$), Let z^+ or z^- be zero. By minimizing z^++z^-, one of z^+ or z^- will be zero.

Rationale: $|x+y-1| = |u - v| \leq |u| + |-v| = u+v$ since $u, v\geq 0$

minimization of $\max\{5, x\}$ $\to$ min v subject to $v\geq 5$, $v-x\geq 0$

Fractional Linear Programming

Consider the fractional linear program $\min\left\{\dfrac{\alpha_0+\alpha_1 x_1..+\alpha_n x_n}{\beta_0+\beta_1 x_1..+\beta_n x_n}\middle| Ax=b; x\geq 0\right\}$ FLP

Assume that ①feasible set of FLP is bounded: $\exists L>0$ with $\|x\|\leq L$ $\forall x$: $Ax=b$, $x\geq 0$

②denominator of objective function is strictly positive $\beta_0+\beta_1 x_1+..\beta_n x_n>0$ for all feasible x.

Homogenization: introduce new variables $y_i\geq 0$, i=1..n and $y_0\geq 0$

Set $x_i=\dfrac{y_i}{y_0}$ (i=1..n), homogenize the fractional linear program as

$$\min\frac{\alpha_0 y_0+\alpha_1 y_1..+\alpha_n y_n}{\beta_0 y_0+\beta_1 y_1..+\beta_n y_n} \quad \text{s.t. } b_i y_0-\sum_{j=1}^{n} a_{ij} y_j=0,\forall i=1..m \quad y_0>0, y_1\geq 0,..y_n\geq 0 \quad \text{HFLP}$$

For any $(y_0,y_1,..,y_n)$ feasible in HFLP, $\lambda(y_0,y_1,..,y_n)$ with $\lambda>0$ is also feasible and has the same objective value.
For each $(y_0,y_1,..,y_n)$, we can then find a λ such that $\beta_0y_0+\beta_1y_1+..\beta_ny_n=1$ and the scaled point will have identical objective. Thus we can restrict our attention to these points and still find an optimal solution.
In other words: The denominator in the objective of HFLP can always be normalized to unity.

Normalized Problem:

$$\min \alpha_0 y_0 + \alpha_1 y_1 .. + \alpha_n y_n$$

$$\text{s.t. } \beta_0 y_0 - \sum_{j=1}^{n} \beta_j y_j = 1 \qquad b_i y_0 - \sum_{j=1}^{n} a_{ij} y_j = 0, \forall i = 1..m \qquad y_0 > 0, y_1 \geq 0, .. y_n \geq 0$$

where the first constraint forces the denominator of the objective of HFLP to be equal to 1. $\Rightarrow$ This is an LP!

Construct a Solution for FLP:

Denote by $(y_0^*,y_1^*,..,y_n^*)$ the optimal solution of the normalized problem.
Then $(y_1^*/y_0^*,.., y_n^*/y_0^*)$ is an optimal solution for FLP. This construction only works if $y_0^*\neq0$.
However, we can show that under the assumptions y_0 cannot be zero at optimality.

Relax the Lower Bound on y_0:

Since the original problem has a bounded feasible set, we can rule out the possibility that $y_0=0$.
Assume to use in HFLP $y_0\geq0$ instead of $y_0>0$.
Suppose $y_0^*=0$ and set $y^*= (y_1^*,...,y_n^*)^T$.
Since $b_iy_0^*=0$, $Ay^*=0$, $y^*\geq0$, Here $y^*\neq0$ since otherwise the denominator of the objective in both HFLP and FLP is not strictly positive as assumed.
If x is feasible in FLP, then $x+\lambda y^*$ is also feasible in FLP $\forall\lambda>0$ since given that $Ay^*=0$, $A(x+\lambda y^*)=Ax+\lambda Ay^*=b$, This would contradict that FLP has a bounded feasible set, so y_0^* cannot be zero and we can use $y_0\geq0$ in HFLP.

Integer Programming

IP: mathematical programming problems where one or more variables are integer valued. (discrete value)
Binary Variables: $x_j\in\{0,1\}$ e.g take "yes" or "no" decisions
Integer Variables: $x_j\in\{0, 1..n\}$ e.g discrete amount, cannot produce 3.6 cars
Program can include both integer and real variables Mixed Integer Linear Programming (MILP)
Application: resource allocation, graph theory, circuit design

MILP and LP relaxed(MILP by dropping integrality constraints) solutions in general are not close.
For minimization, MILP feasible space(points inside feasible polygon) < LP feasible space, f(x* MILP) ≥ f(x* LP).

Common fallacy: “Integer problems are easier to solve than continuous problems”
—Integer problems can be very hard to solve!
—n binary variables define 2^n possible combinations
Integer linear programming is actively researched
—Problems with 1000s of binary variables are solvable —Optimality gaps known for intermediate solutions
Integer non-linear programming still a difficult area
—Heuristics are often needed to aid solvers solve efficiently this classes of models.

Example: Capital Budgeting ≈ Knapsack
—company has resources $i\in\{1,..m\}$, resource i has limited availability bi.
—company can undertake projects $j\in\{1,..n\}$, project j requires aij units of resource i and gives revenue cj.
—which project should be undertaken?
$\max_x z=\sum_{j=1}^{n} c_jx_j$ s.t $\sum_{j=1}^{n} a_{ij}x_j\leq b_i$ $(\forall i\in\{1..m\})$ $x_j\in\{0, 1\}$ $(\forall j\in\{1..n\})$

Example: Facility Location
—company has m potential distribution sites $i\in\{1,..m\}$, build a distribution center at site i cost f_i.
—company has n customers $j\in\{1,..n\}$ whose demands d_j need to be satisfied from one or more distribution centers.
—cij: cost to satisfy an amount xij of customer j’s demand from distribution center i, if center i is built.
—which distribution centres should be build? How should the demand be satisfied to minimize costs?

Example: Airline Crew Scheduling

Combinatorial Optimization problems involve finding a optimal object from a finite set of objects.
A subarea of integer programming
Enumeration gets intractable as problem size grows.
Problems often reducible to few categories: Knapsack problem, Bin-Packing problem, Cutting stock problem, Minimum spanning tree problem…
Special results and algorithms apply to these problems.

Knapsack Problem

Consider n items of weight w_j, j∈{1,..n} and a knapsack of weight capacity W
Item j has value v_j, but not all items may fit the knapsack
To maximize the total value of the knapsack.

$$\max_x z = \sum_{j=1}^{n} v_j x_j \qquad \text{s.t. } \sum_{j=1}^{n} w_j x_j \le W \text{ and } x_j \in \{0,1\} \text{ (whether or not inside the knapsack), } \forall j \in \{1,..n\}$$

Bin-Packing Problem (generalization of knapsack)

n items of weight wj, j∈{1,..n}, k bins of capacity W; xij=1 if item j assigned to bin i, 0 otherwise.
Minimize the number of bins needed to store all items

$$\min_{x,y} \sum_{i=1}^{k} y_i \qquad \text{s.t. } \sum_{j=1}^{n} w_j x_{ij} \le W y_i \text{ and } \sum_{i=1}^{k} x_{ij} = 1, \forall j \in \{1,..n\} \text{ and } x_{ij}, y_i \in \{0,1\}, \forall i \in \{1,..k\}, \forall j \in \{1,..n\}$$

—> each bin is identical, use bins as less as possible, item in exactly one and only one bin

Example: The owner of a big motor company wants to build k=10 new factories in different areas. All factories make the same product. The owner has n=15 customers. Customer i demands di units of the product. The operating cost of the factory j is $f_j \ge 0$ and the maximum number of units it can make is Mj. The cost of delivering 1 unit from factory i to customer j is $c_{i,j}$.
Where should the owner build his new factories in order to minimize the delivery cost + operating cost?
—$y_{i,j}$ denotes the quantity of the product that goes from the factory i to customer j
—xi be the binary variable with xi=1, if i is used, =0, otherwise ∀i=1,..k

$$\min_x \left\{ \sum_{i=1}^{k} \sum_{j=1}^{n} c_{i,j} y_{i,j} + \sum_{i=1}^{k} x_i f_i \right\} \text{ subject to } \sum_{j=1}^{n} y_{i,j} = d_j \quad \sum_{j=1}^{n} y_{i,j} \le M_i x_i \quad y_{i,j} \ge 0 \ \forall i,j \quad x_i \in \{0,1\}$$

Mixed Integer Linear Programming (MILP/MIP, most general class of ILP)

min z=cTx s.t. ①Ax=b and ②$x_j \ge 0$, for j∈N={1,..,n} and ③xj∈$\mathbb{N}_0$={0,1,2..} for j∈Z⊆N

where $x_j \in N \setminus Z$ are continuous, as in LPs Note: RHS b≥0; slack & excess continuous

Specialized Problems:

Pure Integer Linear Programming (Pure ILP): Z=N∪{z}, all variables (including slack and objective value) are integer.
Binary Linear Programming (0-1 ILP): ILP where all variables are binary.
Mixed Integer Binary Programming (MIBP): MILP where integer variables are binary, $x_j \in \{0, 1\}$ for j∈Z.

MILP Standard Form

—Similar to LPs, in particular b ≥ 0.
—Slack and excess variables in MILPs are continuous.

Pure IP standard form

—Slack and excess variables in Pure IPs are integer-valued.
—Step 0. Apply LP standard form transformations, except addition of slack and excess variables, thus
Minimization + Non-negative right-hand-sides + Free variables
—Step 1. Scale the equations of the model so that ***all coefficients are integers***.
—Step 2. Insert ***integer*** slack and/or excess variables.

logic operations→IP

Logical Operator Either-Or

model logical operations on the constraints via integer variables:

$a_1^T x \le b_1 \vee a_2^T x \le b_2$

$a_1^T x \le b_1 + M\delta$
$a_2^T x \le b_2 + M(1-\delta)$
$\delta \in \{0, 1\}$
where M is a large enough constant ("big-M")

If $\delta=1$, it enforces $a_2^T x \le b_2$, but a lot of slacks in $a_1^T x \le b_1 + M\delta$.

Reasonable M: upper bound of x times element-wise max of a1, a2.

Exclusive OR? Same as OR

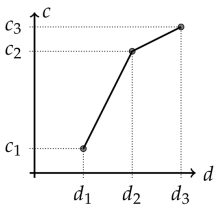
P IMPLIES Q ≡ NOT(P) OR Q

Logical Operator k-out-of-m

Satisfy at least k out of m constraints:
$a_1^T x \le b_1, a_2^T x \le b_2, \ldots, a_m^T x \le b_m$

$a_1^T x \le b_1 + M\delta_1$
$\vdots$
$a_m^T x \le b_m + M\delta_m$
$\sum \delta_j \le m-k$
$\delta_j \in \{0,1\}, \forall j \in \{1,..,m\}$

Example: develop constraints restricting the feasible set to the piecewise linear function



y=a1x+b1 when x∈[d1, d2]
y=a2x+b2 when x∈[d2, d3]

$y \ge a1x+b1 + M\delta$	$-y+a1x \le -b1 + M\delta$
$y \le a1x+b1 - M\delta$	$y-a1x \le b1 + M\delta$
$x \ge d1$	$x \ge d1$
$x \le d2 + M\delta$	$x \le d2 + M\delta$
$y \ge a2x+b2 - M(1-\delta)$	$-y+a2x \le -b2 + M(1-\delta)$
$y \le a2x+b2 + M(1-\delta)$	$y-a2x \le b2 + M(1-\delta)$
$x \ge d2 - M(1-\delta)$	$-x \le -d2 + M(1-\delta)$
$x \le d3$	$x \le d3$

where $\delta \in \{0, 1\}$, $x \in [d1, d3]$, $y \in [c1, c3]$

M is a sufficiently large positive number (upper bound of x times element-wise of maximum of a1 and a2 plus upper bound of y)

Note: 不能 y=(a1x+b1)*δ+(a2x+b2)*(1-δ). Non-linear 了

Finite-Value Variables

Assume a variable x_j can only take a finite number of values: $x_j \in \{p_1,.., p_m\}$.
Introduce variables $z_{j1}, \ldots, z_{jm} \in \{0, 1\}$ and add the constraint $z_{j1}+..+z_{jm}=1$ (*)
Replace $x_j=p_1z_{j1}+..+p_mz_{jm}$ in the objective function and all constraints. Due to (*), x_j can only assume a single value.

Example: $x \in \{1, 3, 11\}$ $x=1+2y_1+10y_2$ $y_1, y_2 \in \{0, 1\}$ $y_1+y_2 \le 1$

How to solve ILPs?
—reuse/extend LP algorithm >> cutting plane algorithm
—define ILP-specific algorithm >> branch-and-bound algorithm

Unimodularity

A unimodular matrix is a matrix whose determinant A is +1 or −1.
A totally unimodular matrix is a matrix for which every square non-singular submatrix is unimodular.
Assume the coefficient matrix A of an ILP to be a m × n matrix whose rows can be partitioned into two disjoint sets B and C. If the following sufficient conditions hold, we say that A is totally unimodular:

- Every column of A contains at most two non-zero entries;
- Every entry in A is 0,+1, or -1;
- If two non-zero entries in a column of A have the same sign, then the row of one is in B, and the other in C;
- If two non-zero entries in a column of A have opposite signs, then the rows of both are in B, or both in C.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

indexes of rows for 2 sets are {1, 2} and {3}

$$\begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

indexes of rows for 2 sets are $\{1, 2, 3, 4\}$ and $\{\varnothing\}$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

indexes of rows for 2 sets are $\{1\}$ and $\{2, 3, 4\}$

A surprising property of ILPs with constraints Ax=b, where A is totally unimodular and x integer valued, is that the solution of the ILP and the solution of its LP relaxation are identical.
This is therefore a special class of ILPs that can be solved using (continuous) LPs.

Cutting Plane Algorithm

LP Relaxation: LP program obtained by replacing all integer variables $x_j \in \mathbb{N}_0$ in a ILP with continuous variables $x_j \in \mathbb{R}$.
LP relaxation has better or same optimal value as ILP.
Outline of solution procedure:

Solve a LP relaxation. (Contains all originally feasible solutions, plus others)
If optimal solution is integer, we are done.
Otherwise, tighten the LP relaxation and repeat.

Tightening: restrict feasible set of the LP relaxation without excluding the optimum solution of the ILP.

Step 0.Write the ILP in standard form (Pure IP standard form)
Step 1.Solve the LP relaxation
Step 2.If the resulting optimal solution x* is integer, STOP; optimal solution found
Step 3.Generate a cut, a constraint satisfied by all feasible integer solutions, but not by a solution x* with non-integer components.
Step 4. Add cut to the LP relaxation and go back to Step 1.
The algorithm terminates after finite number of iterations. The resulting x* is integer and optimal.

Gomory mixed-integer Cuts

Assume x1, ..xn≥0 and integer
Let $\lfloor c \rfloor = \max\{a \in \mathbb{Z}: a \le c\}$ be the floor function. Any real number c can be written as $c = c + (c - \lfloor c \rfloor)$
Construct a Gomory cut for $a_1x_1 + ..+ a_nx_n = b$ where $a_j, b \in \mathbb{R}$ (not necessarily integer)
The constraint can be written as $\left(\lfloor a_1 \rfloor + \underbrace{(a_1 - \lfloor a_1 \rfloor)}_{f_1}\right)x_1 + ..+ \left(\lfloor a_n \rfloor + \underbrace{(a_n - \lfloor a_n \rfloor)}_{f_n}\right)x_n = \left(\lfloor b \rfloor + \underbrace{(b - \lfloor b \rfloor)}_{f}\right)$
Rearrange $f_1x_1 + ..f_nx_n - f = \lfloor b \rfloor - \lfloor a_1 \rfloor x_1 - ..\lfloor a_n \rfloor x_n$

Gomory Cut: assume $x_i \in \mathbb{N}_0$, then $f_1x_1+\ldots+f_nx_n \ge f$ where $f_i=a_i-\lfloor a_i \rfloor$ and $f=b-\lfloor b \rfloor$ (floor)
Note: in practice, we can generate multiple Gomory cuts simultaneously.

Proof: Consider $f_1x_1 + ..f_nx_n - f = \lfloor b \rfloor - \lfloor a_1 \rfloor x_1 - ..\lfloor a_n \rfloor x_n$
—As all $x_i \in \mathbb{N}_0$, right-hand side is integer. Thus LHS must be an integer too.
—Since $x \ge 0$, $1 > f_i \ge 0$, $\forall i$, LHS is lower bounded by -f.
—Since -f>-1, LHS is lower bounded by -1, and given that LHS can only take integer values, it can only be a non-negative integer 0,1,2..
—Therefore, we have proved that $f_1x_1+\ldots+f_nx_n - f \ge 0$

Suppose Step 1 of our cutting plane algorithm gives a non-integer x*.
Then there is a row in the last Simplex tableau that has $x_i^* + \sum_{j \notin I} y_{ij}x_j^* = y_{i0}$ (Row) with $y_{i0} \notin \mathbb{N}_0$.

Note: the summation is on the non-basic variables.

GomoryCut. Setting $f_j=y_{ij}-\lfloor y_{ij} \rfloor$, $f=y_{i0}-\lfloor y_{i0} \rfloor$ $\sum_{j \notin I} f_j x_j \geq f$ (GC)

(GC) is violated by a non-integer x* since $x_j^*=0$ if $j \notin I$, thus $\sum_{j \notin I} f_j x_j^* = 0 < f$

Example:

Gomory cutting plane cuts off the ***current solution***, LP relaxation solution is always infeasible after cutting.

Knapsack Cover Cuts

A set S of items in a knapsack problem is a cover if $\sum_{j \in S} w_j > W$

If S is a cover, then the corresponding knapsack cover cut is $\sum_{j \in S} x_i \leq |S| - 1$

Minimal cover constraint: a cover constraint such that for all proper subsets T of S $\sum_{j \in T} w_i \leq W$

Note: cover cut not perfect, it just add some new information to the problem.

Branch & Bound Algorithm

Mixed Integer Linear Programming (MILP/MIP)

min $z=c^Tx$

s.t. ①$Ax=b$ and ②$x_j \geq 0$, for $j \in N=\{1,..,n\}$ and ③$x_j \in \mathbb{Z}$ for $j \in Z \subseteq N$

where $x_j \in N \setminus Z$ are continuous, as in LPs

Complete Enumeration Algorithm
Divide and Conquer Principle

P_i: i-th subproblem
$x^*(P_i)$: optimal solution for LP relaxation of Pi.
$c^Tx^*(P_i)$: optimal value for LP relaxation of Pi. Set $c^Tx^*(P_i)=\infty$ if Pi is an infeasible LP.
OPT denotes objective function value of best feasible solution (for P_0) found so far. At beginning, OPT=∞.
x_{OPT} is the solution producing the best objective OPT.
A LP solution $x^*(P_i)$ is feasible for MILP P_0 if it satisfies all integrality constraints of P_0.

Branch & Bound Steps for MILPs

1.Initialization
—Initialize OPT=∞.
—Solve LP relaxation of $P_0 \Rightarrow x^*(P_0)$.
—If $x^*(P_0)$ feasible for P_0, OPT = cT $x^*(P_0)$, xOPT = $x^*(P_0)$, and STOP.
Otherwise set list of problems to $\{P_0\}$.
2.Problem Selection: extract from list a P_i having $c^Tx^*(P_i) <$ OPT. If no such P_i exists, STOP.
3.Variable Selection: choose non-integer $x_p^* \in x_p^*(P_i)$ that must be integer in P_0 (i.e. $p \in Z$).
4.Branch: create subproblems P′ and P″ with $x_p \leq \lfloor x_p^*(P_i) \rfloor$ and $x_p \geq \lceil x_p^*(P_i) \rceil$,respectively.
5.Bound P′:
—Solve LP relaxation of P′.
—If $c^Tx^*(P')<$OPT
If $x^*(P')$ feasible for P_0, Set OPT = $c^Tx^*(P')$ and $x_{OPT}=x^*(P')$.
Else add P′ to list for further inspection.
6.Bound P″: (Identical to step 5, but for P″) Go back to step 2.

Output:

OPT=∞: P_0 is infeasible.
OPT<∞: P_0 is feasible and OPT its optimal value.
Optimal Solution: x_{OPT} associated to OPT.
Termination: Under assumption of finite bounds $\underline{x}_j$, $\overline{x}_j$ for j∈Z, algorithm terminates in finitely many steps.

How to chose the Branching variable?
—Choose the most fractional variable, i.e. the one furthest from 0 or 1.
—Develop a heuristic estimating the value in dividing a variable.
—Use strong branching, i.e. try a few branching alternatives and use the best.

Are there methods we could have used to explore this tree more effectively?
There is no guarantee. The worst case complexity stays the same. But, at least for this instance (and in practice for most instances), picking better branching strategies or developing additional cutting planes would have expedited solution time.

Examples: 2个

Branch-and-Cut

MILP solvers typically use a variant of branch-and-bound called branch-and-cut (hybrid algorithm)
- Branch-and-cut combines branch-and-bound and (typically several types of) cutting plane algorithms.
- At each search tree node, cuts are dynamically generated and added to the LP relaxation to tighten the feasible set.
- Cuts are further categorized into local cuts that are valid only for a subtree of the branch-and-bound tree, and global cuts that are valid for all nodes.

Typically outperforms stand-alone methods, e.g. branch-and-bound and cutting plane algorithms.

Duality

Primal Problem: $\max\{c^T x : Ax \le b, x \ge 0\}$ where c, x∈$\mathbb{R}^n$, A∈$\mathbb{R}^{n\times n}$, b∈$\mathbb{R}^m$.

Dual Problem: $\min\{b^T y : Ay \ge c, y \ge 0\}$ where c, A, b are in P, y∈$\mathbb{R}^m$.

Note:
# of constraints → # of variables
# of variables → # of constraints
b → coefficient in objective
Less than or equal to → greater than or equal to
A → transpose of A

Definition is symmetric, dual of D is P.

Weak Duality

Assume problems max{c^Tx: Ax≤b, x≥0} and min{b^Ty: A^Ty≥c, y≥0} are both feasible. Let x∈$\mathbb{R}^n$ be feasible for P and y∈$\mathbb{R}^m$ be feasible for D. Then the objective value $c^Tx \le b^Ty$.
Proof: P requires that Ax≤b, so $y^TAx \le y^Tb$ since y≥0. D requires that $A^Ty \ge c$, so $(A^Ty)^T \ge c^T$ and $(A^Ty)^Tx \ge c^Tx$ since x≥0. Then, if both LPs are feasible, $c^Tx \le y^TAx \le y^Tb = b^Ty$ (since both vector multiplications give a scalar)

Non-Linear Program holds weak duality.

Strong Duality

Assume that problems P and D are both feasible. Let B be optimal basis for P, together with optimal bases solution (x^*_B, x^*_N).
Then, we have ①$y^* = (B^{-1})^T c_B$ is an optimal solution for D
②$c^Tx^* = b^Ty^*$, (optimal value of primal and dual) the objective value coincide.
Proof:

BV	z	$(x_B)^T$	$(x_N)^T$	RHS
z	1	0^T	$-r^T$	$(c_B)^TB^{-1}b$
x_B	0	I	$B^{-1}N$	$B^{-1}b$

If P is unbounded, D is infeasible and vice versa.

Shadow Prices of the primal problem $\Pi = \left(B^{-1}\right)^T c_B$

The optimal solution of the dual problem is $y^*=(B^{-1})^T c_B$.
Thus shadow prices can be obtained by solving the dual.

Non-Linear Program may not hold strong duality.

The simplex algorithm we have seen is often called the primal simplex algorithm.
Start from feasible solution (but suboptimal), then search for optimal feasible solution.
The dual simplex algorithm is similar, but operates on the dual problem.
Strong duality guarantees that the two algorithms return the same optimal solution.

Primal/Dual Possibilities
When the primal is feasible and has a finite optimum, the dual is also feasible and has a finite optimum. When the primal is unbounded, the dual must be infeasible. When the primal is infeasible, the dual is either infeasible or unbounded.

		Primal =feasible + bounded Finite optimal	Unbounded	Infeasible
Dual	Finite optimal	✓		
	Unbounded			✓
	Infeasible		✓	✓

primal no feasible solution & dual no feasible solution:
max 2x1+x2 s.t.-x1+x2≤-4, x1-x2≤2, x1≥0, x2≥0
min -4y1+2y2 s.t. -y1+y2≥2, y1-y2≥1, y1≥0, y2≥0

To Obtain the Dual—

Indirect Way

Bring problem to form of (P) or (D) and apply duality definition.
1. Bring LP to the form of either (P) or (D).
—Replace variables $x_i \in \mathbb{R}$ with $(x_i^+ - x_i^-)$ where $x_i^+, x_i^- \geq 0$.
—Replace equality constraints with two inequality constraints.
—Change constraint direction (≤, ≥) by multiplication with (−1) if necessary.
—Change direction of objective function by multiplication with (−1) if necessary.
2. Obtain dual according to definition.
—If LP is in the form of (P), its dual is(D). If LP is in the form of (D), its dual is(P).
Primal: $\max\left\{c^T x : Ax \leq b, x \geq 0\right\}$ where c, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^m$.
Dual: $\min\left\{b^T y : Ay \geq c, y \geq 0\right\}$ where c, A, b are in P, $y \in \mathbb{R}^m$.
3. Simplify dual problem. (Optional)
—Replace variable pairs $y_i, y_j \geq 0$, $i \neq j$, that occur in all functions as $\alpha y_i - \alpha y_j$ by one variable $y_k \in \mathbb{R}$.
—Replace matching inequality constraints by equality constraints.

Direct Way

Transforming the initial LP to (P) and then obtain (D) can be tedious (so called Indirect way).
Apply duality without detour via (P) or (D).
1.For every primal constraint, create one dual variable. For every primal variable, create one dual constraint.
2.Dual coefficient matrix is A^T. Former right-hand sides b become new costs. Former costs c become new right-hand sides.
3. If primal is max problem: Dual is min problem.

—If i^{th} primal constraint is [≥, =, ≤], i^{th} dual variable becomes [y_i≤0, y_i∈ℝ, y_i≥0], respectively.

—If j^{th} primal variable is [x_j≥0, x_j∈ℝ, x_j≤0], j^{th} dual constraint becomes [≥, =, ≤], respectively.

4. If primal is min problem: Dual is max problem.

—If i^{th} primal constraint is [≥, =, ≤], i^{th} dual variable becomes [y_i≥0, y_i ∈ℝ, y_i≤0], respectively.

—If j^{th} primal variable is [x_j≥0, x_j∈ℝ, x_j≤0], j^{th} dual constraint becomes [≤, =, ≥], respectively.

Note: Equivalence of indirect and direct ways

Example:

Sensitivity Analysis

Perturbation: Let p∈$\mathbb{R}^m$ denote a general RHS and define the value function v(p):$\mathbb{R}^m$→ℝ by v(p)=min{z=c^Tx | Ax=p, x≥0} (m: number of constraints, 1: number of variable)
Solving the original LP (reference problem) min{z=c^Tx | Ax=b, x≥0}, thus computes v(b).
Note that b is the nominal understanding of system.

Suppose we have solved the reference problem min{z=c^Tx | Ax=b, x≥0} and found an optimal basis matrix B satisfying x_B=$B^{-1}b$≥0 (feasibility) and cost vector r=c_N – $N^T(B^{-1})^Tc_B$≥0 (optimality)

The vector of **Shadow Prices** Π∈$\mathbb{R}^m$ is defined as $\Pi = \left(B^{-1}\right)^T c_B$ where B=B(I) is an optimal basis.

Note that there can be more than one optimal basis, the shadow prices need not to be unique.

The shadow prices give information about the sensitivity of the value function v(p) at p=b.

Behaviour of Value Function: $v(p) = v(b) + \Pi^T (p-b)$ for all p∈$\mathbb{R}^m$ with B^{-1}p≥0 (feasibility check), where v(b) is the nominal solution.

Proof: If B^{-1}p≥0, then B remains the optimal basis for min{z=c^Tx | Ax=p, x≥0} since r is not affected by changing b to p. Thus, v(p)=$c_B^TB^{-1}$p=$c_B^TB^{-1}$b+$c_B^TB^{-1}$(p-b)=$c_B^TB^{-1}$b+Π^T(p-b)

Global **Behaviour** of Value Function: v(p) ≥ v(b)+Π^T(p-b) for all p∈$\mathbb{R}^m$ (m constraints). Π: sensitivity

Proof: $v(p)=\min_{x\geq 0, Ax=p}\{c^Tx\}=\min_{x\geq 0, Ax=p}\{c^Tx-\Pi^T(Ax-p)\}$

$\geq \min_{x\geq 0}\{c^Tx-\Pi^T(Ax-p)\}=\min_{x\geq 0}\{(c^T-\Pi^TA)x+\Pi^Tp)\}=\Pi^Tp+\min_{x\geq 0}\{(c^T-\Pi^TA)x\}$

$$\left[c^T-\Pi^TA\right]x=\left(\left[c_B^T\middle|c_N^T\right]-c_B^TB^{-1}\left[B\middle|N\right]\right)\begin{bmatrix}x_B\\x_N\end{bmatrix}=\left[c_B^T\middle|c_N^T\right]\begin{bmatrix}x_B\\x_N\end{bmatrix}-c_B^T\left[I\middle|B^{-1}N\right]\begin{bmatrix}x_B\\x_N\end{bmatrix}=c_B^Tx_B-c_B^Tx_B+\left(c_N^T-c_B^TB^{-1}N\right)x_N=r^Tx_N\geq 0$$

(as r≥0 and xN≥0) So $\min_{x\geq 0}\left\{\left(c^T-\Pi^TA\right)x\right\}\geq 0$

Thus, we find v(p) ≥ Π^Tp +$\min_{x\geq 0}\{(c^T-\Pi^TA)x\}$ ≥ Π^Tp = Π^Tb+Π^T(p-b)= $c_B^TB^{-1}$b+Π^T(p-b) = v(b)+Π^T(p-b)

Assume the company can be additional amount of ingredient X at price μ_1 per unit.
If $\mu_1+\Pi_1<0$ (overall cost decreases), it is worthwhile; Otherwise, overall cost increases.
– Π_1 is the maximum price one should pay for one additional unit of X. (change the availability of X)

New constraints RHS is given as p=b+ξe_t where e_t is a vector with all coordinates 0 except a single 1 at position t.
Accept offer → Total production cost v(b)+μ_tξ

Extra production → Total production cost $\begin{cases} = v(b)+\Pi_t\xi, \ if\ B^{-1}(b+\xi e_t)\geq 0 \\ \geq v(b)+\Pi_t\xi, \ in\ general \end{cases}$

ACCEPT offer if $\mu_t+\Pi_t<0$ and if B^{-1}(b+ξe_t)≥0; REJECT offer if $\mu_t+\Pi_t>0$ and if B^{-1}(b+ξe_t)≥0
Π_t is the maximum price one should pay.

For Maximization, (Local) if B^{-1}p≥0, then v(p)=v(b)+Π^T(p-b)

(Global) $v(p) \le v(b)+\Pi^T(p-b)$ for all $p \in \mathbb{R}^m$.

Suppose row t is initially a "≤ constraint" and a slack variable x_s had been added. Then $\Pi_t=\beta_s$ where β_s is the objective coefficient of x_s in the final (optimal) tableau. — shadow prices can be read off the final tableau

Proof:

If x_s is non basic in the final tableau, then

$$\beta_s = -r_s = -(r_t)^T e_s = -(c_N - N^T(B^{-1})^T c_B)^T e_s = -c_N^T e_s + (N^T\Pi)^T e_s = -c_s + \Pi^T a_s = 0 + \Pi^T e_t = \Pi_t$$

where es is a vector of zeros expect for a 1 in the s^{th} position, $a_s=Ne_s$ is the column s of A; Since xs is the slack for row t, we noted that $c_s=0$ and $a_s=e_t$.

If x_s is basic in the final tableau, then $\beta_s=0=c_s=(e_s)^T c_B=(e_s)^T B^T\Pi=(e_t)^T\Pi=\Pi_t$

Suppose row t is initially a "≥ constraint" and a surplus variable x_s had been added. Then $\Pi_t=-\beta_s$ where β_s is the objective coefficient of x_s in the final (optimal) tableau.

Basic Representation

$$\begin{aligned} z-\left(c_N-N^TB^{-T}c_B\right)^T x_N &= c_B^T B^{-1} b \\ x_B + B^{-1}Nx_N &= B^{-1}b \end{aligned}$$

Tableau

BV	z	$(x_B)^T$	$(x_N)^T$	RHS
z	1	0^T	$-r^T$	$(c_B)^T B^{-1} b$
x_B	0	I	$B^{-1}N$	$B^{-1}b$

where I is m×m identity matrix.

General Tableau Notation (for a feasible index set I with $p \in I$, $q \notin I$)

BV	z	x_1	..	x_p	..	x_q	..	x_n	RHS
z	1	β_1	..	β_p	..	β_q	..	β_n	β_0
:	:	:		:		:		:	:
x_p	0	y_{p1}	..	y_{pp}	..	y_{pq}	..	y_{pn}	y_{p0}
:	:	:		:		:		:	:

Note: $y_{ii}=1\ \forall i \in I$ and $y_{ji}=0\ \forall i \in I, j \in I\setminus\{i\}$ $\beta_i=-r_i\ \forall i \in /I, i \neq 0$ (negative reduced cost) $\beta_i=0\ \forall i \in I$

Game Theory: Zero Sum Game, Pure Strategies & Mixed Strategies

Game theory is a important application of Duality.

LP/ILP study problems with a single decision-maker.
Game Theory studies problems with multiple decision-maker.

- —pricing policy in an oligopoly.
- —campaigning strategy in presidential election
- —strategy in military battles

Games
(key properties)outcome depends on combination of strategies selected by all players
players must guess what other players will do

Two-Person Zero-Sum Game:

two players: row player (RP) and column player (CP)
RP chooses one out of m strategies (row strategies)
CP chooses one out of n strategies (column strategies)
Zero-Sum: RP wins whatever CP loses, and vice versa

Payoff Matrix: descriptor of a two-player zero-sum game
If RP plays strategy i and CP plays strategy j, then CP pays a_{ij} to RP

		CP			
		Strategy 1	Strategy 2	…	Strategy n
RP	Strategy 1	a11	a12		a1n
	Strategy 2	a21			

RP	…				
	Strategy m	am1			amn

Example: Odds-Evens
Example: Rock-Paper-Scissors

Assumptions of Two-Person Zero-Sum Games:
—each player knows the game setting (available strategies to RP and CP, values of payoff matrix)
—Both players simultaneously choose their strategy, without knowing what their opponent chooses
—Each player chooses a strategy that enables him/her to do best, reasoning as if the opponent could anticipate his/her strategy
—Both players are rational: They try to maximize their utility; They show no compassion for their opponent

Example: elections game

Dominance
Dominated Row Strategy: row strategy i is dominated by row strategy i′ if $a_{i'j} \geq a_{ij}$ for all column strategies j=1,..n and $a_{i'j} > a_{ij}$ for some j
Note: regard as there is one constraint inactive
Dominated Column Strategy: column strategy j is dominated by column strategy j′ if $a_{i'j} \leq a_{ij}$ for all row strategies i=1,..m and $a_{i'j} < a_{ij}$ for some i
Note: A rational player will never play a dominated strategy. A rational opponent knows this.

Dominant Strategy Equilibrium: if a repeated removal of dominated strategies leads to a game where each player has just one strategy left, then this strategy pair is a dominant strategy equilibrium.
Properties:
If a dominant strategy equilibrium exists, then it is unique.
If a dominant strategy equilibrium exists, then any rational players will play the associated equilibrium strategies.

Minimum Row Payoff
Assumption: "Each player chooses a strategy that enables him/her to do best, reasoning as if the opponent could anticipate his/her strategy"
—αi: expected payoff of a row strategy i, i.e., the payoff of the RP if row strategy i is chosen
—RP cannot be stopped by CP from playing i.
—RP reasons as if CP could anticipate RP's move.
—Given that the RP chooses i, the CP will pick the strategy j that minimizes the CP loss, thus $\alpha_i = \min_{j=1,..n} a_{ij}$

Maximum Column Payoff
We repeat the reasoning for the CP.
—βj: expected payoff of a column strategy j, i.e., the payoff of the columns player if column strategy j is chosen
—CP cannot be stopped by RP from playing j.
—CP reasons as if RP could anticipate CP's move.
—Therefore, given that the CP chooses j, the RP will pick the strategy i that maximizes his payoff, thus: $\beta_j = \max_{i=1,..m} a_{ij}$

A Nash Equilibrium is a strategy pair such that no player has an incentive to unilaterally deviate from his/her chosen strategy if told the strategy of the other player.
A **Nash Equilibrium** in pure strategies is thus achieved if: max αi = min βj
where αi and βj are payoffs for row strategy i and column strategy j, with i =1,..m j =1,..n.
Properties:
—A Nash equilibrium may not always exist in pure strategies.
—The payoff of the Nash equilibrium's strategy pair is the value of the game.

Mixed Strategies

pure strategy: choose 1 strategy all of the time

Example: Odds-and-Evens
In a mixed strategy (p_1,.. p_m; q_1,.. q_n):

—RP plays strategy i with probability p_i , $\sum_{i=1}^{m} p_i = 1$

—CP plays strategy j with probability q_j , $\sum_{j=1}^{n} q_j = 1$

—If p_k=1or q_k=1, then k is a pure strategy

—The payoff of the mixed strategy will be $V = \sum_{i=1}^{m}\sum_{j=1}^{n} p_i q_j a_{ij}$

—RP seeks probabilities that maximize payoff (p_1*,.. p_m*) CP seeks probabilities that minimize loss (q_1*,.. q_n*)

Column Player Perspective

Assumption: "Each player chooses a strategy that enables him/her to do best, reasoning as if the opponent could anticipate his/her strategy"
—CP expects RP to respond with optimal p_i 's for any choice of q_j's. How should CP choose the q_j's?

$$V_{CP} = \min_{q_1,..q_n} \max_{p_1,..p_m} \sum_{i=1}^{m}\sum_{j=1}^{n} p_i q_j a_{ij} \quad \text{subject to } \sum_{j=1}^{n} q_j = 1 \quad q_j \geq 0 \quad \text{and } \sum_{i=1}^{m} p_i = 1 \quad p_i \geq 0$$

Optimization problem is non-linear due to the $p_i q_j$ terms
We study the formulation as nested programs

$$V_{CP} = \min_{q_1,..q_n} V_{CP}^{in}(q_1,..q_n) \qquad \text{subject to } \sum_{j=1}^{n} q_j = 1 \quad q_j \geq 0$$

For any choice of q_j's, let $\alpha_i = \sum_{j=1}^{n} q_j a_{ij}$ be row payoffs

Then the inner maximization problem is $V_{CP}^{in}(q_1,..q_n) = \max_{p_1,..p_m} \sum_{i=1}^{m} p_i \alpha_i$ subject to $\sum_{i=1}^{m} p_i = 1 \quad p_i \geq 0$

The solution is p_i=1 for the largest α_i, p_k=0 for k≠i.

The inner maximization optimal value is thus simply $V_{CP}^{in}(q_1,..q_n) = \max\{\alpha_1,..\alpha_n\}$
Expanding the definitions of the α_i's, we conclude that CP is in fact solving a min-max problem

$$V_{CP} = \min_{q_1,..q_n} \max \left\{ \sum_{j=1}^{n} q_j a_{1j}, .. \sum_{j=1}^{n} q_j a_{mj} \right\} \text{ subject to } \sum_{j=1}^{n} q_j = 1 \quad q_j \geq 0$$

The min-max problem is equivalent to

$$\text{a linear program } V_{CP} = \min_{\tau, q_1,..q_n} \tau \text{ subject to } \tau \geq \sum_{j=1}^{n} q_j a_{ij} \qquad (\forall i=1..m) \sum_{j=1}^{n} q_j = 1 \quad q_j \geq 0$$

Note: the optimal q_j^*'s are independent of the p_i^*'s

Row Player Perspective

A similar reasoning applies to the row player, who instead optimizes

$$V_{RP} = \max_{p_1,..p_m} \min_{q_1,..q_n} \sum_{i=1}^{m}\sum_{j=1}^{n} p_i q_j a_{ij} \quad \text{subject to } \sum_{i=1}^{m} p_i = 1 \quad p_i \geq 0 \text{ and } \sum_{j=1}^{n} q_j = 1 \quad q_j \geq 0$$

The max-min problem can be shown equivalent to a linear program

$$\text{a linear program } V_{RP} = \max_{\tau, p_1,..p_m} \tau \text{ subject to } \tau \leq \sum_{j=1}^{m} p_i a_{ij} \qquad (\forall j=1..n) \qquad \sum_{i=1}^{m} p_i = 1 \qquad p_i \geq 0$$

Note: p_i^*'s will be independent of the q_j^*'s

Minimax Theorem

For every two-person zero-sum game, the RP and CP linear programs have the same optimal value $V^*=V^*_{RP}=V^*_{CP}$

Proof (outline): result follows by strong duality since the two programs are the duals of each other.

Consequences:

—A Nash Equilibrium in mixed strategies always exists

A Nash Equilibrium is **a strategy pair** such that no player has an incentive to unilaterally deviate from his/her chosen strategy if **told** the strategy of the other player.

—Players expect identical payoffs

—Neither player has an incentive to change p_i or q_j

Statement generalizes to M players (Nash, 1951).